

The Bazaar Approach to Physics Education

Seth Rosenberg [UMd/CCNY]
Dick Furnstahl [The OSU]

- Problems with “conventional” PER-based reform
→ particularly for upper-level courses
- Development models for computer software:
“The Cathedral and the Bazaar”
 - evolvable systems (cf. physics research!)
 - importance of open-source philosophy
 - implementation (sociology)
- Tools for Bazaar-style development
- Prototype project

- A recent project at Ohio State funded by the NSF attempted to introduce “reform” elements into an upper-level course for physics majors previously taught in a traditional, pure-lecture format. These reforms included using a studio classroom and greatly increasing the active component of lectures (such as with informal groupwork).
- The PI was Dick Furnstahl, an active researcher in nuclear theory. [Apologies for Furnstahl’s absence from AAPT meeting: quarter system at OSU and 2 classes 5 days a week!] Seth Rosenberg helped with development and assessment of the project.
- While there have been positive outcomes from the project, there have also been many difficulties and frustrations in trying to develop, implement, and assess the reform curriculum. These problems are common to many physics education research and development projects but are particularly acute in upper-level courses.
- Here we propose analogies with computer software development models. A comparison of two extremes was made by Eric Raymond, who labeled them the cathedral and the bazaar (note the spelling!) [RA1]. We associate conventional textbook-development and PER approaches with the cathedral and argue here for more bazaar-style development.
- We will present an overview and examples. An emphasis is on the proven tools for open-source software development that might be taken over for PER development.

Limitations of PER: Upper Level Courses

- Cycle time is long
- Alpha/beta testing population is very small
- Limited and/or outdated connections to contemporary research
- The need to reinvent wheels
- Unrecognized subtleties in the physics
- Curriculum development and assessment is a huge time sink for an active research physicist
- Community acceptance of PER

Conventional approaches to curriculum development, implementation, and assessment have many limitations, particularly when applied to upper division courses.

- **Cycle time is very long**
 - for upper level courses, it is typically a full year to revisit the same content → comparable time between successive trials of an experiment or assessment.
 - many years to produce *publishable* results in form of tutorials, text, or papers (by conventional standards)
- **Alpha/beta testing population is very small**
 - often just local physics majors → small numbers
- **Limited and/or outdated connections to contemporary research**
 - materials tend to provide snapshots of research at best, and then are static (with infrequent revisions at best)
 - this is particularly unfortunate given the wealth of accessible modern experiments (e.g., amo QM expts.)
 - but more extensive connections requires interaction with researchers (typically absent from development of curriculum)

- The need to reinvent wheels
 - particularly clear with software-based work: we wanted a student feedback system that could accommodate equations. None of the existing “products” on the market could do that, but we couldn’t build on them because they were “closed source”. Had to start entirely from scratch rather than adding value to existing products.
 - having to reinvent wheels hinders innovation by erecting a large barrier to participation by most of the people who could make significant contributions
 - claim: applies to non-software materials (e.g., tutorials) as well
- Recognizing subtleties in the physics
 - quantum mechanics, for example, is tricky! Researchers generating curriculum materials often miss subtleties or miss opportunities for better explanations. *The best physics is in texts by the best physicists (e.g., Purcell)*.
 - the difficulty here is the small number of developers (conventional beta testing is too inefficient and incomplete and happens too late in the development)
- Curriculum development and assessment is a huge time sink for an active research physicist
 - discourages efforts → waste of expertise!
 - or else very limited (or one-time) assessment is done

- Community acceptance of PER
 - when conventional physics “researchers” are not involved in PER, they tend not to believe it and resist implementing it, particularly since there is usually an increased time commitment.
 - waiting until rock solid PER results are available before implementation is too slow and not consistent with effective physics research, which is incremental (complete and correct theories do not appear at once) and characterized by parallel efforts and shared results.
- Bottom line: There is a tremendous pool of talent that is not being tapped. Ok, what do we propose to do?
Time for some analogies!

Two Models for Software Development

- *The Cathedral and the Bazaar* [Eric Raymond]
 - Cathedral → closed source, proprietary development (e.g., Microsoft)
 - Bazaar → open-source development model
 - * massively parallel development
 - * keywords: openness, peer review, free (as in speech, not beer) software
- Development environments
 - Cathedral mode:
 - “... carefully crafted by individual wizards or a small band of mages working in splendid isolation, with no beta to be released before its time.”

- Eric Raymond's essay "The Cathedral and the Bazaar" [RA1] provides a starting point for a discussion of competing models of software development. One can take issue with various details in this and other Raymond essays but they providing a useful overview for this discussion.
 - The difference between open and closed source involves the freedom ("free as in speech, not beer") to examine, modify, and incorporate elsewhere a piece of software. This is essential for large-scale peer review and massively parallel development.
 - One should note here and throughout the discussion the similarities of the bazaar mode and effective physics research (note also that there is also much cathedral building in physics, which often hinders progress!).
- Cathedral mode always involves beta testing, but in a very restricted form. Beta testers are not treated as co-developers. Cathedral beta testers merely point at problems; open source alpha and beta testers offer *fixes* (which are evaluated and incorporated).

- Development environments (cont.)

- Bazaar mode:

- “... great babbling bazaar of differing agendas and approaches . . . out of which a coherent and stable system could seemingly emerge only by a succession of miracles.”

- Advantage of evolvable systems

- “Centrally designed protocols start out strong and improve *logarithmically*. Evolvable protocols start out weak and improve *exponentially*.”

- “Only solutions that produce partial results when partially implemented can succeed. Evolvable systems begin partially working right away and then grow, rather than needing to be perfected and frozen.”

- Contributors to the bazaar come in with “differing agendas and approaches” and many different motivations (like the physics community!).
 - An important feature of successful projects (more later) is that while most participants are in a great babbling bazaar, the final decisions are more coherently reached by a small group (sometimes one person).
- The advantages of evolvable systems are obvious. The question is how to achieve them. The claim is that cathedral-style development is not well suited to create evolvable systems, but bazaar-style development is.
 - growth of web is good example [SH1]
 - decision to include “View source . . .” in first browsers made html code of *any* page open so anyone could emulate and build on page layouts or techniques → lowered barriers to new pages → enormous pace of design development.
(Ask yourself: would Microsoft have included “View source . . .” if they had led the way?)
 - also essential to evolvability: separation of the software engine (which does the work) from the user interface (which does the viewing and control). In contrast, software with tight integration between creation, file format, and display (such as Excel or Lotus Notes) does not easily evolve.

Examples of Open Source Projects

- **Cathedral:** Microsoft Windows 2000 and Office, Adobe Illustrator, Mathematica, etc.
 - note: these are very good products!
- **Bazaar:** Apache, sendmail, Gnu/Linux, Perl, Python, PHP, Samba, MySQL, BIND, . . .
 - Apache runs ~ 50% of world's web servers
 - Perl is behind much WWW "live content"
 - BIND provides domain name service for 'net
 - sendmail is the main email transport software
 - plus thousands of smaller scale projects
- **High quality:** many are "category killers"
- **Complex:** e.g, Linux (complete operating system)

- Many (or most) of the familiar software products on the market are proprietary, closed source, and were built mostly cathedral-style. The argument here is *not* that you cannot develop excellent products this way, but that the bazaar offers an alternative that has many compelling features.
 - For example, innovation is enhanced.
 - MIT's Technology Review recently compiled a list of 100 young innovators in science, technology, and the arts [MIT1]. Selected as innovator of the year was Miguel de Icaza of Mexico City for his work on the open-source GNOME project. Also on the list: Linus Torvald (Linux czar). Conspicuously absent: anyone from Microsoft!
- There is a long list of open source software tools and applications. The quality is very high because of the open source market environment and the nature of the hacker culture. See [OR1].
 - Many of these are “category killers”: not only extremely capable and robust, but so good that no commercial competition has challenged them (e.g., BIND and sendmail)
- The projects are not restricted to simple programs. The GNU/Linux operating system is an example of a complex system developed in bazaar mode. It consists of the Linux kernel plus hundreds of open-source packages.
 - Linux has evolved rapidly: it started in 1991 as a hobby of graduate student Linus Torvald, who wanted a unix-like operating system for his 386 PC.

Parallels: PER and Software Development

- Examples of Cathedral vs. Bazaar in PER
 - Cathedral → ordinary textbook development, conventional PER model
 - Bazaar → none as yet (although many projects have this flavor to some degree)
- Development community
 - Cathedral: Microsoft → textbook authors, PERG members
 - Bazaar: Hackers → Physics teachers (and students!) at all levels
 - * large (potential) pool of expertise
 - * willingness to contribute “in spare time”

- Before going further, we'll make explicit the parallels we have in mind between physics education curriculum development and software development.
 - cathedral, closed-source development is the usual mode for textbooks but also for physics education research, development and dissemination projects. Our claim is that these have the characteristics of centrally designed protocols: they start out strong but only improve logarithmically.
 - we are unaware of any projects that are really in the bazaar mode, although many have some aspects (such as open source)
 - we can argue endlessly about whether this characterization is fair, but this would not be constructive compared to actually trying to run a project in bazaar mode. We propose such a project below.
- The development community is clear for the cathedral. Our claim is that for the bazaar there is a strong parallel to the hacker community, which are the physics teachers (and some students).
 - “in spare time” means teaching for many research professors!

- Example: Technical Support
 - Cathedral: Pay the vendor for support → support from textbook authors, PERG experts
 - * in principle, reliable support, since “paid for”
 - * in practice, support can be uneven or very limited or have a slow turnaround
 - Bazaar: The Internet! → tap the resources of the physics community
 - * in principle, unreliable since no one in charge or directly accountable
 - * in practice, tremendous resources tapped by web search engines
 - * Infoworld gave its “Best Technical Support Award” to “Linux people on the Internet”

- One can develop the parallels further, but here we just cite an example: technical support of a product. Cathedral support comes from the vendor and you pay for it (often as part of the purchase price). Bazaar support comes from the community.
 - Linux support is incredible. Almost any arcane problem or topic: someone has encountered it *and* there are posted resolutions. If not, post yourself and see the amazingly fast turnaround time.

Bazaar-Mode Lessons (Raymond)

Some maxims for open-source development

→ from Raymond's experience and observations

→ each is applicable to PER!

1. Every good work of software starts by scratching a developer's personal itch.
2. Good programmers know what to write. Great ones know what to rewrite (and reuse).
3. "Plan to throw one away; you will, anyhow."
4. If you have the right attitude, interesting problems will find you.
5. When you lose interest in a program, your last duty to it is to hand it off to a competent successor.
6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
7. Release early. Release often. And listen to your customers.

There are many important characteristics of bazaar-mode development. To give you a flavor, we consider some of the maxims Eric Raymond presents (the ones in red) and how they carry over to PER and curriculum development and reform [RA1].

1. E.g., Linux started with Linus Torvald's itch for a simple 386 OS. Open source ("free software") is essential here. Otherwise the barriers are too high with too much reinventing of the wheel. But if you can start from someone else's work and go in your own direction (like physics research!), then it works.
5. This is an essential part of the culture. The project is the focus, not the developer. There are many examples of open-source projects being "handed off".
6. This is a difference from the ordinary use of beta testers.
7. It is essential to release early and often. But working on the bleeding edge is not for everyone. Since many users do not want buggy versions that change daily, one separates "developer's" versions from "stable" releases (there is a standard version numbering system that keeps track of these distinctions).

Bazaar-Mode Lessons (Raymond) [cont.]

8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone (else).
9. Smart data structures and dumb code works a lot better than the other way around.
10. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource.
11. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.
12. Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
13. "Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away."
14. Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected.

8. This is a great advantage of massively parallel development. Note that the problem identification and the best fix will very often come from different people.
10. Empirical observation: People *will* make contributions of their valuable time if they are treated right.
11. This is how advisors stay in business! The most original ideas frequently come from the young and inexperienced, but one must be able to separate the wheat from the chaff.

Implementation of PER-Based Bazaars

- Successful bazaar projects don't run themselves
 - bazaar characterizes dynamics of contributors
 - control at the top is essential to provide quality assurance and to avoid fragmentation
 - “hacker” culture → constraints
- Implementation models for bazaar development
 - Choose from successful models in open-source software development
 - E.g., benevolent dictator (Linux) or voting council (Apache)
 - Difficulties in maintaining a consistent style necessitate a dictator or small council

- One might be misled by the description of the bazaar into thinking that one just tosses an idea and some source code out onto the net and a complicated project builds and maintains itself. *This is not the case!* Successful bazaar projects don't run themselves, but are (usually) tightly controlled at the top by the project coordinator(s) [CO1].
- The “bazaar” aspect of the development is in the dynamics of the contributors to the project, who are outside of the control of the coordinators. They can do what they want based on their own motivations. (In a more global sense, the interplay of different projects is also bazaar-like.) Of course, one of their choices, if they don't like the direction the project is taking, is to split off (“fork”) with their own version (or, less drastically, tailor it to their own needs).
- The leadership at the top has to provide the quality assurance to prevent excessive forking and fragmentation. In practice, there are constraints from the hacker culture (“hacker” is not a derogatory term, by the way, and should not be confused with “cracker”), see below.
- How is the top-level control implemented? There are several successful models, ranging from benevolent dictatorships to rotating dictatorships to constitutional monarchies to ruling councils (see references linked in [RA1]).
 - There are typically two tiers of contributors as well: ordinary contributors and co-developers. The co-developers have responsibility for major subsystems.

- **Critical importance of open source**
 - Stallman: “free as in speech, not as in beer”
 - computer code but also curriculum materials
 - materials must be reproducible at all stages, not just in polished, published form
- **Role of PERG members?**
 - initiators and coordinators of “Open Source” PER-based projects
- **Why not just rely on new journal(s)?**
 - pace of conventional PER is glacial
 - participation from very small fraction of physics community
 - threshold for “publishability” is too high
 - * too complete a product or result is required

- The success of bazaar-style development depends critically on having open source.
 - This is also known as “free software,” which is often misinterpreted as meaning non-commercial (no cost). In fact, “free” in this context refers to the freedom one has to read, modify, and incorporate the code. Richard Stallman, the mastermind behind the GNU project (he also wrote emacs!), stresses that it is “free as in speech, not as in beer” and that commercial distributions are fine, as long as the code is available [FSF1]. For example, Redhat sells packaged versions of Linux, all of which could be downloaded from the net without charge. (That being said, most free software is, in fact, free of charge.)
 - As applied to physics education development, this means not only that software is open, but all materials are open and can not only be used, but adapted and incorporated (with proper attribution) or tailored to local environments. The “agreement” is that improvements are fed back to the coordinator.
 - This is just like in physics research, where new ideas can be taken and extended without constraint. In many instances, research is *not* open, as when codes or raw data are not released or the description of a calculation is insufficient to allow the reader to reproduce it completely. **In such cases, the pace of innovation and development is dramatically slowed!**

- What about members of physics education research groups?
 - They are ideally suited to be the initiators and/or coordinators of open source projects based on physics education research.
 - Credit is not a problem: the coordinator tends to receive credit far beyond his/her individual contributions! (Translating this into documentation for a tenure case *is possible*.)
- As a side note, some comments on the new journal(s) that have been started to provide outlets for physics education research:
 - they serve a vital function, but are not sufficient for the type of development we describe
 - the pace is slow and participation is low (from the large community of physics teachers and researchers)
 - most importantly, the threshold for participation is simply much too high. Because of (necessarily) high standards, it takes much too long and requires too much time and effort for anyone who is not a dedicated physics education researcher. The “masses” are excluded!
 - one might argue that this is true of any physics specialty, but PER *curriculum development* is different in this respect.
[*This argument needs to be fleshed out!*]

- How do you make money?
 - figurative (how do PE researchers get tenure?)
 - literal (how do publishers stay in business?)
 - funding based on peer-reviewed outcomes

- So if everything is free how does one make money? One can ask this question in both figurative and literal senses.
 - Figuratively, the general payoff for contributing to an open source project is in reputation (just as in physics research!).
 - * in tenure cases, this is documented in the letters from experts in the field.
 - * the bean counters will want some definite metrics, however (e.g., number of publications or citations).
 - Literally, there is the question of how publishers fit into the equation.
 - * while there are business models for making money with open source projects, it is too early to judge whether they are viable
 - How should government agencies (e.g., NSF-DUE) fund open-source projects?
 - * fund the coordinator(s)
 - * evaluate based on peer-reviewed outcomes, as usual

Pre-Rebuttals

A pre-emptive attack on some potential criticisms:

1. But curriculum materials *are* “open source” by construction! (I.e., anyone can read them.)
2. What about intellectual property rights?
3. Why should someone spend time producing materials that anyone can copy?
4. What about quality control?
5. Won't there be a million versions of everything, many of them wrong, untested, etc.?
6. Is this approach is so great, why wasn't it used before?

1. Just because the materials can be read doesn't mean they are open source. One needs: the source code (TeX or Word files that can be edited directly) and *permission* to change or extend the material and use/distribute it (with appropriate restrictions → open-source licensing). Much effort has gone into developing appropriate licensing for this process [LI1].
2. The culture and the licensing are geared to maintain credit for intellectual developments. Copyrights are held by the authors but you cannot charge license fees. Removing a person's name from a project history is absolutely not done without consent. The treatment of intellectual property is similar to that for theoretical ideas in physics.
3. cf. physics → different utility function. Reputation is the coin!
4. The examples from the open source community show that quality control can be excellent in bazaar development. The testing and quality assurance is in the process (massive peer review) and the control at the top.

5. Several issues here:

- The *possibility* of code “forks” are essential because the globally best path to take may not be clear locally. The historical experience of forking in software development has pluses and minuses. Most of the minuses were associated with proprietary versions, such as the fractionation of unices in the 1980’s. But the pluses are that different paths can be explored and then complete to determine the best one (needed to find global minima!). Examples are glibc vs. libc (big fight over C libraries that eventually was won by glibc), the different linux distributions (which take turns leapfrogging each other), and KDE vs. Gnome.
- Not everything will work optimally so one should rely on competition in the “market” to find the most effective solution(s) (or to develop different variations for different environments).
- In practice, forking and fragmentation in open-source software is suppressed by unwritten community rules that “assign control of an open-source project, including the right to designate ‘official’ versions, to a single entity (an individual, an informal group, or a formal organization).” Also, there are strong social pressures against forking.
- “Rogue” versions have not been a problem because of public review and because there is typically a single place to get an “official” version that has undergone additional review and testing.

6. The maturation of the internet has led to qualitatively new possibilities for collaborative development. New tools are essential, however (next!).

Tools for Bazaar-Style Development

- **Why now?** Because of internet + new tools for massively parallel collaborative development
 - Example of new technology changing how physicists work: Los Alamos preprint server
 - Here: some tools that can be adapted to physics education research
 - * Bugzilla — bug-tracking system
 - * Faq-O-Matic — “knowledge base” system
 - * SourceForge — free hosting service

- Sample tools for massively parallel collaborative development. Qualitative difference because of quantitative change in tools.
- The Los Alamos server has dramatically changed the pace of development in some subfields of physics as well as permanently altered the role of journals. A preprint in nuclear physics effective field theory will often have 2/3 of its references to unpublished Los Alamos preprints and there may be several preprint cycles before journal articles appear. Different types of contributions (like summer school lectures) can carry equal weight. No obvious problems with lack of explicit refereeing; instead there is (implicit) widespread peer review (cf. Amazon.com book reviews or “open content”).
- See [BU1] for a long list of “Call Center, Bug Tracking, and Project Management Tools for Linux.”

Tools for the Bazaar [cont.]

- Bugzilla (<http://www.mozilla.org/projects/bugzilla/>)
 - sample bug-tracking system
 - A database for bugs that lets people report bugs via the web and then assigns them to appropriate developers
 - “bug” is a generic term for typos, software bugs, requests for enhancements, suggestions for changes
 - Bugzilla priorities bugs, coordinates schedules (“milestones”), maintains to-do lists, and tracks bug dependencies
 - Sample application: tracking errata and updates to curriculum materials

- Keeping track of “bugs” is essential.
 - note that “bugs” also include requests for enhancements
- Bugzilla is a database for bugs.
 - also called a “Defect Tracking System”
 - it lets people report bugs and assigns these bugs to the appropriate developers.
 - developers can use bugzilla to keep a to-do list as well as to prioritize, schedule, and track dependencies.
 - far superior to shared lists and email.
 - see “The Bugzilla Guide Home Page” (<http://www.trilobyte.net/barnsons/>) for more detail, especially the section “Why Should We Use Bugzilla” (<http://www.trilobyte.net/barnsons/html/why.html>)
- Directly adaptable to curriculum materials!
 - Every text should use this type of tool!

Tools for the Bazaar [cont.]

- [Faq-O-Matic \(faqomatic.sourceforge.net\)](http://faqomatic.sourceforge.net)
 - “knowledge base” system
 - a web-based system that automates an FAQ (Frequently Asked Questions list)
 - highly searchable database of “questions”
 - anyone can contribute
 - * permission system to establish hierarchy of moderators
 - Sample application: supplement to curriculum materials
 - See also: “open content” documentation [e.g., Andamooka (www.andamooka.org) or php manual (<http://www.php.net/manual>)]

- FAQ's are great but difficult to maintain. Faq-O-Matic solves the problem!
- “Open content” documentation “. . . opens books up to contribution, comment, and criticism via section-by-section community annotation and makes the book (including annotation) available for download.” Coordinated by David Sweet, recent physics Ph.D. from U.Maryland, now a quantitative analyst for a hedge fund.

- SourceForge (<http://sourceforge.net>)
 - Free hosting service for open-source projects
 - Features: CVS repository, mailing lists, bug tracking, message boards/forums, task management software, site hosting, permanent file archival, full backups
 - * CVS → Concurrent Version System
 - A tool to keep track of changes made by developers working on the same files.
 - * total web-based, secure administration
 - * to set up a project, register as a site user, login, and register your project. That's it!
 - SourceForge is itself an open source project
 - * started to remove obstacles to open source software development
 - * "A small idea that refused to stop growing."
 - Use to host open-source curriculum projects!

- Extensive documentation is available at <http://sourceforge.net>; see the documentation page http://sourceforge.net/docman/?group_id=1

Prototype Bazaar Project

- A common PE “itch” → physics simulations
- Simulation software → natural starting point
 - use existing development tools and strategies
- Modern version of CUPS (and similar projects)
 - relativity, E&M, quantum mechanics, . . .
 - implement pedagogy from PER as part of PER research and development cycle
 - build in assessment and feedback tools

- We finish up by sketching a proposed bazaar project that will (hopefully) go online later this year, coordinated initially by Dick Furnstahl (and anyone else he can convince).
- The first of Raymond's maxims was that every good work of software starts by scratching a developer's personal itch. A common itch in upper-level physics education is for good physics software simulations.
- A development project for simulation software is a natural starting point for bringing the bazaar to physics education
 - can build on past efforts, such as CUPS, which had some elements of the bazaar
 - existing development tools and strategies are *designed* for software development
 - provides testbed for open source PE development and dissemination
- The basic idea is to produce a modern, PER-based, open source version of the CUPS simulations [CU1]
 - cover all branches of physics in modular form
 - implement pedagogy from PER in the design of the simulation interfaces and in the associated materials as part of a PER research and development cycle
 - build in tools to permit “easy” assessment and feedback to the instructor and to the project

- Features:

- decouple hardware implementation (including graphics) from interface from simulation routines from associated curriculum materials
- platform independence
- downloadable (and updates) from the web
 - * entirely open source

- Essential initial conditions:

- a critical mass of initial code is needed
- can start off crude, buggy, incomplete, poorly documented but must have plausible promise
- coordinator has to recognize (and use) good design ideas from contributors
- coordinator sets vision rather than managing contributors → “the best innovators aren’t dictated to: they are turned loose”

- Features:
 - As with web browsers, the key to evolvability is to decouple the hardware implementation from the interface *and* from the simulation routines themselves *and* from the associated curriculum materials
 - * “decoupled” means that these elements communicate through a well-defined, standardized protocol
 - platform independence is important for wide dissemination
 - downloadable (including all updates) from the web, entirely open source (of course!)
- There are some essential initial conditions for a viable project.
 - There has to be sufficient code that works somewhat to show where things are going and that the project is feasible. It doesn't have to be perfect! Plausible promise is the key.
 - In addition, the coordinator(s) must be able and willing to recognize and incorporate good ideas from contributors. The coordinator cannot dictate to contributors, but must set a vision for the project (and make contributors feel that their efforts are worthwhile). [SH1]

- Potential development community:
 - physics teachers at all levels who would like to use the simulations in their classes
 - physics researchers simulating their specialty
 - * cf. Physics 2000 simulations
[<http://www.colorado.edu/physics/2000/>]
 - undergraduate students doing research project
→ great source of computer expertise!
 - hackers interested in physics
 - * **large** community (just read slashdot)
 - some subset → co-developers (e.g., in charge of individual modules)

- The potential development community is extensive.
 - physics teachers at all levels who would like to use the simulations in their classes
 - physics researchers simulating their specialty, such as the Physics 2000 simulations from the University of Colorado
 - undergraduate research projects
 - great source of computer expertise!
 - hackers interested in physics; this is a large community that is apparent by browsing slashdot (“News for Nerds”)
 - some subset of these contributors would become co-developers of various aspects of the project

Summary

- Bazaar-style development addresses:
 - short cycle times → parallel efforts
 - dissemination → same time as development
 - reinventing → reuse/extend existing materials
 - static content → continuous updates
 - time sink → lower barriers to contribute
 - physics subtleties → involve experts
 - community acceptance → get them involved!
- Tools for massively parallel collaborative development are readily available
- Physics education curriculum development and reform have largely operated in Cathedral mode; it is time to try the Bazaar!

- In summary, bazaar-style development addresses all of the limitations of physics education research for upper level courses:
 - parallel efforts at many different institutions alleviate the short cycle times, extend the alpha/beta populations, and build in large scale dissemination
 - open source means reuse rather than reinvention and, together with the development philosophy and culture, dramatically lowers the barriers for contributors (as well as institutionalizing continuous updates)
 - with the involvement of more physicists, bugs become shallow and modern pedagogy becomes better tested and accepted in the community
- The basic tools that enable massively parallel collaborative development are available → It's time to try the Bazaar!

References

We have avoided associating dates with most of these references, which are still evolving. Some of these references have not been explicitly cited in the text, but all contributed ideas to the talk.

- [BR1] Christopher Brown, "Linux and Decentralized Development,"
http://www.firstmonday.dk/issues/issue3_3/browne/ or
<http://vip.hex.net/~cbbrowne/lfs.html>

- [BU1] "Call Center, Bug Tracking, and Project Management Tools for Linux," <http://linas.org/linux/pm.html>.

- [CO1] Charles Connell, "Open Source Projects Manage Themselves? Dream On,"
<http://www.lotus.com/developers/devbase.nsf/articles/doc200009>
and the response from Eric Raymond,
<http://www.lotus.com/developers/devbase.nsf/articles/doc200009>

- [CU1] CUPS is the Consortium for Upper-Level Physics Software, which is a series of book/software packages with computer simulations on a wide range of physics topics. They are DOS programs written in Turbo Pascal. See <http://www.physics.gmu.edu/~cups/> for details.

- [FSF1] Richard Stallman is the driving force behind the Free Software Foundation. See <http://www.fsf.org/philosophy/philosophy.html> for an extensive set of links to the philosophy of free software and information about the GNU Project.

- [HE1] Frank Hecker, "Setting Up Shop: The Business of Open-Source Software,"
<http://www.hecker.org/writings/setting-up-shop.html>
- [KA1] Dan Kaminsky, "Core Competencies: Why Open Source is the Optimum Economic Paradigm for Software,"
<http://doxpara.netpedia.net/core.html>
- [KU1] Ko Kuwabara, "Linux: A Bazaar at the Edge of Chaos,"
http://firstmonday.org/issues/issue5_3/kuwabara/index.html.
- [LE1] Josh Lerner and Jean Triole, "The Simple Economics of Open Source," <http://papers.nber.org/papers/w7600>.
- [LI1] See <http://www.opensource.org/licenses/index.html> for descriptions of the most common open-source licenses.
- [LK1] See <http://www.alt.net/~lk/cathedral-bazaar.html>
- [MA1] Malcolm Maclachlan, "Panelists Describe Open Source Dictatorships,"
<http://www.informationweek.com/story/TWB19990812S0003>
- [MIT1] Technology Review, November/December 1999. See <http://www.techreview.com/magazine/tr100/index.asp> and <http://www.techreview.com/magazine/tr100/IOTYpr.asp> for details.
- [OR1] Tim O'Reilly, "The Open-Source Revolution," in <http://release1.edventure.com/Issues/1198.pdf>
- [RA1] Eric Raymond, "The Cathedral and the Bazaar" and related essays. The quotes in the talk are taken from this

essay. See

<http://www.tuxedo.org/~esr/writings/cathedral-bazaar/> for the current text and links to commentary and criticism. Other aspects of the open-source process and culture are discussed in “Homesteading the Noosphere” and “The Magic Cauldron” (available on the same page).

[SH1] Clay Shirky, “View Source... Lessons from the Web’s massively parallel development,”

http://www.shirky.com/OpenSource/view_source.html

and “In Praise of Evolvable Systems,”

<http://www.shirky.com/OpenSource/evolve.html>.

[ST1] Mitch Stoltz, “The Case for Government Promotion of Open Source Software,”

<http://www.netaction.org/opensrc/oss-report.html>.

[ZY1] Con Zymaris, “Shoulders of Giants — A Paper on the Inevitability of Open Source Dominance,”

<http://www.cyber.com.au/users/conz/shoulders.html>